**A program For Photographers Event**

**Introduction**

ABC is a company based in Toronto for organizing events, currently it works on a project to organize an event for 5 days for a group of photographers from all over the world.The company has to put a good program, including a hotel of residence, a hall for meetings, places of landscape to visit, stores for shopping, restaurants and cafes. So the company’s purpose is to make a list of places of landscape in Toronto, including the nearest restaurants, cafes, and shopping stores for each place.

**Data Description**

The data used in this project is provided by Foursquare location data. The data are grouped by landscape area, and each area included the information about this area and all information about restaurants, cafes, and stores which in this area.

**Table of contents**

**01- Import Libraries**

**02- Define Foursquare Credentials**

**03- Define the city and get its latitude & longitude**

**04- Search for Hotels & clean dataframe**

**05- Search for Parks & clean dataframe**

**06- Search for Restaurants & clean dataframe**

**07- Search for Cafeteria & clean dataframe**

**08- Search for Shopping Stores & clean dataframe**

**09- Generate map to visualize hotels, shopping stores and Cafeteria and how they cluster together**

**10- Generate map to visualize Park, Restaurant and Cafeteria and how they cluster together**

**Import Libraries**

In [1]:

**import** **requests** *# to handle requests*

**import** **pandas** **as** **pd** *# for data analsysis*

**import** **numpy** **as** **np** *# to handle data in a vectorized manner*

!conda install -c conda-forge geopy --yes

**from** **geopy.geocoders** **import** Nominatim *# module to convert an address into latitude and longitude values*

*# libraries for displaying images*

**from** **IPython.display** **import** Image

**from** **IPython.core.display** **import** HTML

*#tranforming json file into a pandas dataframe library*

**from** **pandas.io.json** **import** json\_normalize

!conda install -c conda-forge folium=0.5.0 --yes

**import** **folium** *# plotting library*

**Define Foursquare Credentials**

In [2]:

ClIENT\_ID = '4FQQJE2LMGVLERRRE2BOFR1LQQR5ZEHFLTD1WRAFKOFT4TI0' *# your Foursquare ID*

ClIENT\_SECRET = 'RCSNR5TH31NQRG2QYCHK101HRMTUZHBB2TSJ1CJU5UOGRXRQ' *# your Foursquare Secret*

VERSION = '20180604'

LIMIT =30

print('Your credentails:')

print('Foursquare\_ID: ' + ClIENT\_ID)

print('Foursquare\_Secret:' + ClIENT\_SECRET)

**Define the city and get its latitude & longitude**

In [3]:

*# define the city and get its latitude & longitude*

city = 'Toronto'

geolocator = Nominatim(user\_agent="foursquare\_agent")

location = geolocator.geocode(city)

latitude = location.latitude

longitude = location.longitude

print(latitude, longitude)

**Search for Hotels**

In [4]:

*# search for hotels*

search\_query = 'Hotel'

radius = 500

*# Define the corresponding URL*

url = 'https://api.foursquare.com/v2/venues/search?client\_id=**{}**&client\_secret=**{}**&ll=**{}**,**{}**&v=**{}**&query=**{}**&radius=**{}**&limit=**{}**'\

.format(ClIENT\_ID, ClIENT\_SECRET, latitude, longitude, VERSION, search\_query, radius, LIMIT)

url

In [5]:

*# Send the GET Request and examine the results*

results = requests.get(url).json()

*#results*

In [6]:

*# assign relevant part of JSON to venues*

venues = results['response']['venues']

*# tranform venues into a dataframe*

dataframe = json\_normalize(venues)

dataframe.head()

**Clean Hotel Dataframe**

In [7]:

*# keep only columns that include venue name, and anything that is associated with location*

clean\_columns = ['name', 'categories'] + [col **for** col **in** dataframe.columns **if** col.startswith('location.')]+ ['id']

clean\_dataframe = dataframe.loc[:,clean\_columns]

*# function that extracts the category of the venue*

**def** get\_category\_type(row):

**try**:

categories\_list = row['categories']

**except**:

categories\_list = row['venue.categories']

**if** len(categories\_list) == 0:

**return** **None**

**else**:

**return** categories\_list[0]['name']

*# filter the category for each row*

clean\_dataframe['categories'] = clean\_dataframe.apply(get\_category\_type, axis=1)

*# clean column names by keeping only last term*

clean\_dataframe.columns = [column.split('.')[-1] **for** column **in** clean\_dataframe.columns]

clean\_dataframe.head()

In [8]:

*# delete unnecessary columns*

clean\_dataframe2= clean\_dataframe.drop(['cc', 'city', 'country', 'crossStreet', 'distance', 'formattedAddress',\

'labeledLatLngs','neighborhood', 'id'], axis=1)

clean\_dataframe2

In [9]:

*# delete rows with none values*

clean\_dataframe3 = clean\_dataframe2.dropna(axis=0, how='any', thresh=**None**, subset=**None**, inplace=**False**)

clean\_dataframe3

In [10]:

*# delete rows which its category is not Hotel or Event Space*

array= ['Hotel', 'Event Space']

hotel\_dataframe= clean\_dataframe3.loc[clean\_dataframe3['categories'].isin(array)]

hotel\_dataframe

In [11]:

*# delete rows which has duplicate hotel's name*

df\_hotels = hotel\_dataframe.drop\_duplicates(subset='name', keep="first")

df\_hotels

In [12]:

*# choose the hotel which has the same postalCode with the event space*

df\_hotel = df\_hotels[df\_hotels.postalCode == 'M5H 2M9']

df\_hotel

**Search for Parks**

In [13]:

*# search for Parks*

search\_query = 'Park'

radius = 10000

*# Define the corresponding URL*

url = 'https://api.foursquare.com/v2/venues/search?client\_id=**{}**&client\_secret=**{}**&ll=**{}**,**{}**&v=**{}**&query=**{}**&radius=**{}**&limit=**{}**'\

.format(ClIENT\_ID, ClIENT\_SECRET, latitude, longitude, VERSION, search\_query, radius, LIMIT)

url

In [14]:

*# Send the GET Request and examine the results*

presults = requests.get(url).json()

*#presults*

In [15]:

*# assign relevant part of JSON to venues*

venues = presults['response']['venues']

*# tranform venues into a dataframe*

park\_dataframe = json\_normalize(venues)

park\_dataframe.head()

**Clean Park Dataframe**

In [16]:

*# keep only columns that include venue name, and anything that is associated with location*

park\_clean\_columns = ['name', 'categories'] + [col **for** col **in** park\_dataframe.columns **if** col.startswith('location.')]+ ['id']

clean\_park\_dataframe = park\_dataframe.loc[:,park\_clean\_columns]

*# function that extracts the category of the venue*

**def** get\_category\_type(row):

**try**:

categories\_list1 = row['categories']

**except**:

categories\_list1 = row['venue.categories']

**if** len(categories\_list1) == 0:

**return** **None**

**else**:

**return** categories\_list1[0]['name']

*# filter the category for each row*

clean\_park\_dataframe['categories'] = clean\_park\_dataframe.apply(get\_category\_type, axis=1)

*# clean column names by keeping only last term*

clean\_park\_dataframe.columns = [column.split('.')[-1] **for** column **in** clean\_park\_dataframe.columns]

clean\_park\_dataframe.head()

In [17]:

*# delete unnecessary columns*

clean\_park\_dataframe2= clean\_park\_dataframe.drop(['cc', 'city', 'country', 'crossStreet', 'distance', 'formattedAddress',\

'labeledLatLngs', 'id'], axis=1)

clean\_park\_dataframe2

In [18]:

*# delete rows with none values*

clean\_park\_dataframe3 = clean\_park\_dataframe2.dropna(axis=0, how='any', thresh=**None**, subset=**None**, inplace=**False**)

clean\_park\_dataframe3

In [19]:

*# delete rows which its category is not Park*

df\_park = clean\_park\_dataframe3[clean\_park\_dataframe3.categories == 'Park']

df\_park

**Search for Restaurants**

In [20]:

*# search for Restaurants*

search\_query = 'Restaurant'

radius = 10000

*# Define the corresponding URL*

url = 'https://api.foursquare.com/v2/venues/search?client\_id=**{}**&client\_secret=**{}**&ll=**{}**,**{}**&v=**{}**&query=**{}**&radius=**{}**&limit=**{}**'.format(ClIENT\_ID, ClIENT\_SECRET, latitude, longitude, VERSION, search\_query, radius, LIMIT)

url

In [21]:

*# Send the GET Request and examine the results*

Rresults = requests.get(url).json()

*#Rresults*

In [22]:

*# assign relevant part of JSON to venues*

venues = Rresults['response']['venues']

*# tranform venues into a dataframe*

Restaurant\_dataframe = json\_normalize(venues)

Restaurant\_dataframe.head()

**Clean Restaurant Dataframe**

In [23]:

*# keep only columns that include venue name, and anything that is associated with location*

Restaurant\_clean\_columns = ['name', 'categories'] + [col **for** col **in** Restaurant\_dataframe.columns **if** col.startswith('location.')]+ ['id']

clean\_Restaurant\_dataframe = Restaurant\_dataframe.loc[:,Restaurant\_clean\_columns]

*# function that extracts the category of the venue*

**def** get\_category\_type(row):

**try**:

categories\_list3 = row['categories']

**except**:

categories\_list3 = row['venue.categories']

**if** len(categories\_list3) == 0:

**return** **None**

**else**:

**return** categories\_list3[0]['name']

*# filter the category for each row*

clean\_Restaurant\_dataframe['categories'] = clean\_Restaurant\_dataframe.apply(get\_category\_type, axis=1)

*# clean column names by keeping only last term*

clean\_Restaurant\_dataframe.columns = [column.split('.')[-1] **for** column **in** clean\_Restaurant\_dataframe.columns]

clean\_Restaurant\_dataframe.head()

In [24]:

*# delete unnecessary columns*

clean\_Restaurant\_dataframe2= clean\_Restaurant\_dataframe.drop(['cc', 'city', 'country', 'crossStreet', 'distance', 'formattedAddress',\

'labeledLatLngs', 'neighborhood', 'id'], axis=1)

clean\_Restaurant\_dataframe2

In [25]:

*# delete rows with none values*

df\_Restaurant = clean\_Restaurant\_dataframe2.dropna(axis=0, how='any', thresh=**None**, subset=**None**, inplace=**False**)

df\_Restaurant

**Search for Cafeteria**

In [26]:

*# search for Cafeteria*

search\_query = 'Cafeteria'

radius = 10000

*# Define the corresponding URL*

url = 'https://api.foursquare.com/v2/venues/search?client\_id=**{}**&client\_secret=**{}**&ll=**{}**,**{}**&v=**{}**&query=**{}**&radius=**{}**&limit=**{}**'.format(ClIENT\_ID, ClIENT\_SECRET, latitude, longitude, VERSION, search\_query, radius, LIMIT)

url

Out[26]:

'https://api.foursquare.com/v2/venues/search?client\_id=4FQQJE2LMGVLERRRE2BOFR1LQQR5ZEHFLTD1WRAFKOFT4TI0&client\_secret=RCSNR5TH31NQRG2QYCHK101HRMTUZHBB2TSJ1CJU5UOGRXRQ&ll=43.653963,-79.387207&v=20180604&query=Cafeteria&radius=10000&limit=30'

In [27]:

*# Send the GET Request and examine the results*

cresults = requests.get(url).json()

*#cresults*

In [28]:

*# assign relevant part of JSON to venues*

venues = cresults['response']['venues']

*# tranform venues into a dataframe*

Cafeteria\_dataframe = json\_normalize(venues)

Cafeteria\_dataframe.head()

**Clean Cafeteria Dataframe**

In [29]:

*# keep only columns that include venue name, and anything that is associated with location*

Cafeteria\_clean\_columns = ['name', 'categories'] + [col **for** col **in** Cafeteria\_dataframe.columns **if** col.startswith('location.')]+ ['id']

clean\_Cafeteria\_dataframe = Cafeteria\_dataframe.loc[:,Cafeteria\_clean\_columns]

*# function that extracts the category of the venue*

**def** get\_category\_type(row):

**try**:

categories\_list4 = row['categories']

**except**:

categories\_list4 = row['venue.categories']

**if** len(categories\_list4) == 0:

**return** **None**

**else**:

**return** categories\_list4[0]['name']

*# filter the category for each row*

clean\_Cafeteria\_dataframe['categories'] = clean\_Cafeteria\_dataframe.apply(get\_category\_type, axis=1)

*# clean column names by keeping only last term*

clean\_Cafeteria\_dataframe.columns = [column.split('.')[-1] **for** column **in** clean\_Cafeteria\_dataframe.columns]

clean\_Cafeteria\_dataframe.head()

In [30]:

*# delete unnecessary columns*

clean\_Cafeteria\_dataframe2= clean\_Cafeteria\_dataframe.drop(['cc', 'city', 'country', 'crossStreet', 'distance', 'formattedAddress',\

'labeledLatLngs', 'id'], axis=1)

clean\_Cafeteria\_dataframe2

Out[30]:

In [31]:

*# delete rows with none values*

df\_Cafeteria = clean\_Cafeteria\_dataframe2.dropna(axis=0, how='any', thresh=**None**, subset=**None**, inplace=**False**)

df\_Cafeteria

**Search for Shopping Stores**

In [32]:

*# search for Shopping*

search\_query = 'Shopping'

radius = 1000

*# Define the corresponding URL*

url = 'https://api.foursquare.com/v2/venues/search?client\_id=**{}**&client\_secret=**{}**&ll=**{}**,**{}**&v=**{}**&query=**{}**&radius=**{}**&limit=**{}**'.format(ClIENT\_ID, ClIENT\_SECRET, latitude, longitude, VERSION, search\_query, radius, LIMIT)

url

Out[32]:

'https://api.foursquare.com/v2/venues/search?client\_id=4FQQJE2LMGVLERRRE2BOFR1LQQR5ZEHFLTD1WRAFKOFT4TI0&client\_secret=RCSNR5TH31NQRG2QYCHK101HRMTUZHBB2TSJ1CJU5UOGRXRQ&ll=43.653963,-79.387207&v=20180604&query=Shopping&radius=1000&limit=30'

In [33]:

*# Send the GET Request and examine the results*

sresults = requests.get(url).json()

*#sresults*

In [34]:

*# assign relevant part of JSON to venues*

venues = sresults['response']['venues']

*# tranform venues into a dataframe*

Shopping\_dataframe = json\_normalize(venues)

Shopping\_dataframe.head()

**Clean Shopping Dataframe**

In [35]:

*# keep only columns that include venue name, and anything that is associated with location*

Shopping\_clean\_columns = ['name', 'categories'] + [col **for** col **in** Shopping\_dataframe.columns **if** col.startswith('location.')]+ ['id']

clean\_Shopping\_dataframe = Shopping\_dataframe.loc[:,Shopping\_clean\_columns]

*# function that extracts the category of the venue*

**def** get\_category\_type(row):

**try**:

categories\_list5 = row['categories']

**except**:

categories\_list5 = row['venue.categories']

**if** len(categories\_list5) == 0:

**return** **None**

**else**:

**return** categories\_list5[0]['name']

*# filter the category for each row*

clean\_Shopping\_dataframe['categories'] = clean\_Shopping\_dataframe.apply(get\_category\_type, axis=1)

*# clean column names by keeping only last term*

clean\_Shopping\_dataframe.columns = [column.split('.')[-1] **for** column **in** clean\_Shopping\_dataframe.columns]

clean\_Shopping\_dataframe.head()

In [36]:

*# delete unnecessary columns*

clean\_Shopping\_dataframe2= clean\_Shopping\_dataframe.drop(['cc', 'city', 'country', 'crossStreet', 'distance', 'formattedAddress',\

'labeledLatLngs', 'neighborhood', 'id'], axis=1)

clean\_Shopping\_dataframe2

In [37]:

*# delete rows which its category is not Shopping Mall*

df\_Shopping = clean\_Shopping\_dataframe2[clean\_Shopping\_dataframe2.categories == 'Shopping Mall']

df\_Shopping

**Generate map to visualize hotels, shopping stores and Cafeteria and how they cluster together**

In [38]:

*# create dataframe of hotels, shopping stores and Cafeteria*

hotel\_neighbourhood\_df = pd.concat([df\_hotel, df\_Cafeteria, df\_Shopping], ignore\_index=**True**)

hotel\_neighbourhood\_df

In [39]:

*# Generate map to visualize hotel neighbourhood including shopping stores and Cafeteria*

hotel\_map = folium.Map(location=[latitude, longitude], zoom\_start=14)

**for** lat, lng, name, categories, address **in** zip(hotel\_neighbourhood\_df['lat'], hotel\_neighbourhood\_df['lng'],

hotel\_neighbourhood\_df['name'], hotel\_neighbourhood\_df['categories'],\

hotel\_neighbourhood\_df['address']):

label = '**{}**, **{}**'.format(name, address)

label = folium.Popup(label, parse\_html=**True**)

folium.CircleMarker(

[lat, lng],

radius=5,

popup=label,

color='blue',

fill=**True**,

fill\_color='blue',

fill\_opacity=0.7,

parse\_html=**False**).add\_to(hotel\_map)

hotel\_map

**Generate map to visualize Park, Restaurant and Cafeteria and how they cluster together**

In [40]:

*# create dataframe of Park, Restaurant and Cafeteria*

park\_neighbourhood\_df = pd.concat([df\_park, df\_Restaurant, df\_Cafeteria,], ignore\_index=**True**)

park\_neighbourhood\_df

In [41]:

*# Generate map to visualize park neighbourhood including Restaurant and Cafeteria*

park\_map = folium.Map(location=[latitude, longitude], zoom\_start=14)

**for** lat, lng, name, categories, address **in** zip(park\_neighbourhood\_df['lat'], park\_neighbourhood\_df['lng'],

park\_neighbourhood\_df['name'], park\_neighbourhood\_df['categories'],\

park\_neighbourhood\_df['address']):

label = '**{}**, **{}**'.format(name, address)

label = folium.Popup(label, parse\_html=**True**)

folium.CircleMarker(

[lat, lng],

radius=5,

popup=label,

color='blue',

fill=**True**,

fill\_color='blue',

fill\_opacity=0.7,

parse\_html=**False**).add\_to(park\_map)

park\_map